Stream: Internet Engineering Task Force (IETF)
RFC: 8926
Category: Standards Track
Published: September 2020
ISSN: 2070-1721
Authors: J. Gross, Ed. (*Intel*); I. Ganga, Ed. (*Intel*); T. Sridhar, Ed. (*VMware*)

RFC 8926
Geneve: Generic Network Virtualization Encapsulation

Abstract

Network virtualization involves the cooperation of devices with a wide variety of capabilities such as software and hardware tunnel endpoints, transit fabrics, and centralized control clusters. As a result of their role in tying together different elements in the system, the requirements on tunnels are influenced by all of these components. Therefore, flexibility is the most important aspect of a tunnel protocol if it is to keep pace with the evolution of the system. This document describes Geneve, an encapsulation protocol designed to recognize and accommodate these changing capabilities and needs.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8926.

Copyright Notice

Table of Contents

1. Introduction

Networking has long featured a variety of tunneling, tagging, and other encapsulation mechanisms. However, the advent of network virtualization has caused a surge of renewed interest and a corresponding increase in the introduction of new protocols. The large number of protocols in this space -- for example, ranging all the way from VLANs [IEEE.802.1Q_2018] and MPLS [RFC3031] through the more recent VXLAN (Virtual eXtensible Local Area Network) [RFC7348] and NVGRE (Network Virtualization Using Generic Routing Encapsulation) [RFC7637] -- often leads to questions about the need for new encapsulation formats and what it is about network virtualization in particular that leads to their proliferation. Note that the list of protocols presented above is non-exhaustive.

While many encapsulation protocols seek to simply partition the underlay network or bridge between two domains, network virtualization views the transit network as providing connectivity between multiple components of a distributed system. In many ways, this system is similar to a chassis switch with the IP underlay network playing the role of the backplane and tunnel endpoints on the edge as line cards. When viewed in this light, the requirements placed on the tunnel protocol are significantly different in terms of the quantity of metadata necessary and the role of transit nodes.

Work such as "VL2: A Scalable and Flexible Data Center Network" [VL2] and "NVO3 Data Plane Requirements" [NVO3-DATAPLANE] have described some of the properties that the data plane must have to support network virtualization. However, one additional defining requirement is the need to carry metadata (e.g., system state) along with the packet data; example use cases of metadata are noted below. The use of some metadata is certainly not a foreign concept -- nearly all protocols used for network virtualization have at least 24 bits of identifier space as a way to partition between tenants. This is often described as overcoming the limits of 12-bit VLANs; when seen in that context or any context where it is a true tenant identifier, 16 million possible entries is a large number. However, the reality is that the metadata is not exclusively used to identify tenants, and encoding other information quickly starts to crowd the space. In fact, when compared to the tags used to exchange metadata between line cards on a chassis switch, 24-bit identifiers start to look quite small. There are nearly endless uses for this metadata, ranging from storing input port identifiers for simple security policies to sending service-based context for advanced middlebox applications that terminate and re-encapsulate Geneve traffic.

Existing tunnel protocols have each attempted to solve different aspects of these new requirements only to be quickly rendered out of date by changing control-plane implementations and advancements. Furthermore, software and hardware components and controllers all have different advantages and rates of evolution -- a fact that should be viewed as a benefit, not a liability or limitation. This document describes Geneve, a protocol that seeks to avoid these problems by providing a framework for tunneling for network virtualization rather than being prescriptive about the entire system.

1.1. Requirements Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

The Network Virtualization over Layer 3 (NVO3) Framework [RFC7365] defines many of the concepts commonly used in network virtualization. In addition, the following terms are specifically meaningful in this document:

Checksum offload:

An optimization implemented by many NICs (Network Interface Controllers) that enables computation and verification of upper-layer protocol checksums in hardware on transmit and receive, respectively. This typically includes IP and TCP/UDP checksums that would otherwise be computed by the protocol stack in software.

Clos network: A technique for composing network fabrics larger than a single switch while maintaining non-blocking bandwidth across connection points. ECMP is used to divide traffic across the multiple links and switches that constitute the fabric. Sometimes termed "leaf and spine" or "fat tree" topologies.

ECMP: Equal Cost Multipath. A routing mechanism for selecting from among multiple best next-hop paths by hashing packet headers in order to better utilize network bandwidth while avoiding reordering of packets within a flow.

Geneve: Generic Network Virtualization Encapsulation. The tunnel protocol described in this document.

LRO: Large Receive Offload. The receive-side equivalent function of LSO, in which multiple protocol segments (primarily TCP) are coalesced into larger data units.

LSO: Large Segmentation Offload. A function provided by many commercial NICs that allows data units larger than the MTU to be passed to the NIC to improve performance, the NIC being responsible for creating smaller segments of a size less than or equal to the MTU with correct protocol headers. When referring specifically to TCP/IP, this feature is often known as TSO (TCP Segmentation Offload).

Middlebox: The term "middlebox" refers to network service functions (NSFs) or service interposition appliances that typically implement Network Virtualization Edge (NVE) functionality, which terminates or re-encapsulates Geneve traffic.

NIC: Network Interface Controller. Also called "Network Interface Card" or "Network Adapter". A NIC could be part of a tunnel endpoint or transit device and can either process or aid in the processing of Geneve packets.

Transit device: A forwarding element (e.g., router or switch) along the path of the tunnel making up part of the underlay network. A transit device may be capable of understanding the Geneve packet format but does not originate or terminate Geneve packets.

Tunnel endpoint: A component performing encapsulation and decapsulation of packets, such as Ethernet frames or IP datagrams, in Geneve headers. As the ultimate consumer of any tunnel metadata, tunnel endpoints have the highest level of requirements for parsing and interpreting tunnel headers. Tunnel endpoints may consist of either software or hardware implementations or a combination of the two. Tunnel endpoints are frequently a component of an NVE (Network Virtualization Edge) but may also be found in middleboxes or other elements making up an NVO3 Network.

VM: Virtual Machine.

2. Design Requirements

Geneve is designed to support network virtualization use cases for data-center environments, where tunnels are typically established to act as a backplane between the virtual switches residing in hypervisors, physical switches, or middleboxes or other appliances. An arbitrary IP network can be used as an underlay, although Clos networks composed using ECMP links are a common choice to provide consistent bisectional bandwidth across all connection points. Many of the concepts of network virtualization overlays over Layer 3 IP networks are described in the NVO3 Framework [RFC7365]. Figure 1 shows an example of a hypervisor, a top-of-rack switch for connectivity to physical servers, and a WAN uplink connected using Geneve tunnels over a simplified Clos network. These tunnels are used to encapsulate and forward frames from the attached components, such as VMs or physical links.

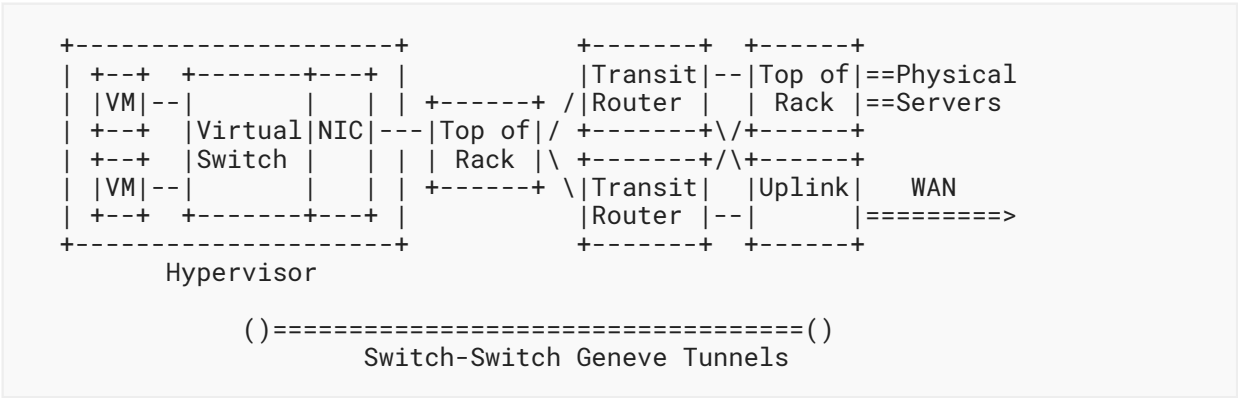


Figure 1: Sample Geneve Deployment

To support the needs of network virtualization, the tunnel protocol should be able to take advantage of the differing (and evolving) capabilities of each type of device in both the underlay and overlay networks. This results in the following requirements being placed on the data-plane tunneling protocol:

- The data plane is generic and extensible enough to support current and future control planes.
- Tunnel components are efficiently implementable in both hardware and software without restricting capabilities to the lowest common denominator.
- High performance over existing IP fabrics.

These requirements are described further in the following subsections.

2.1. Control-Plane Independence

Although some protocols for network virtualization have included a control plane as part of the tunnel format specification (most notably, VXLAN [RFC7348] prescribed a multicast learning-based control plane), these specifications have largely been treated as describing only the data format. The VXLAN packet format has actually seen a wide variety of control planes built on top of it.

There is a clear advantage in settling on a data format: most of the protocols are only superficially different and there is little advantage in duplicating effort. However, the same cannot be said of control planes, which are diverse in very fundamental ways. The case for standardization is also less clear given the wide variety in requirements, goals, and deployment scenarios.

As a result of this reality, Geneve is a pure tunnel format specification that is capable of fulfilling the needs of many control planes by explicitly not selecting any one of them. This simultaneously promotes a shared data format and reduces the chance of obsolescence by future control-plane enhancements.

2.2. Data-Plane Extensibility

Achieving the level of flexibility needed to support current and future control planes effectively requires an options infrastructure to allow new metadata types to be defined, deployed, and either finalized or retired. Options also allow for differentiation of products by encouraging independent development in each vendor's core specialty, leading to an overall faster pace of advancement. By far, the most common mechanism for implementing options is the Type-Length-Value (TLV) format.

It should be noted that while options can be used to support non-wirespeed control packets, they are equally important on data packets as well to segregate and direct forwarding (for instance, the examples given before of input port-based security policies and terminating/re-encapsulating service interposition both require tags to be placed on data packets). Therefore, while it would be desirable to limit the extensibility to only control packets for the purposes of simplifying the datapath, that would not satisfy the design requirements.

2.2.1. Efficient Implementation

There is often a conflict between software flexibility and hardware performance that is difficult to resolve. For a given set of functionality, it is obviously desirable to maximize performance. However, that does not mean new features that cannot be run at a desired speed today should be disallowed. Therefore, for a protocol to be efficiently implementable, a set of common capabilities can be reasonably handled across platforms, along with a graceful mechanism to handle more advanced features in the appropriate situations.

The use of a variable-length header and options in a protocol often raises questions about whether it is truly efficiently implementable in hardware. To answer this question in the context of Geneve, it is important to first divide "hardware" into two categories: tunnel endpoints and transit devices.

Tunnel endpoints must be able to parse the variable header, including any options, and take action. Since these devices are actively participating in the protocol, they are the most affected by Geneve. However, as tunnel endpoints are the ultimate consumers of the data, transmitters can tailor their output to the capabilities of the recipient.

Transit devices may be able to interpret the options; however, as non-terminating devices, transit devices do not originate or terminate the Geneve packet. Hence, they **MUST NOT** modify Geneve headers and **MUST NOT** insert or delete options, which is the responsibility of tunnel endpoints. Options, if present in the packet, **MUST** only be generated and terminated by tunnel endpoints. The participation of transit devices in interpreting options is **OPTIONAL**.

Further, either tunnel endpoints or transit devices **MAY** use offload capabilities of NICs, such as checksum offload, to improve the performance of Geneve packet processing. The presence of a Geneve variable-length header should not prevent the tunnel endpoints and transit devices from using such offload capabilities.

2.3. Use of Standard IP Fabrics

IP has clearly cemented its place as the dominant transport mechanism, and many techniques have evolved over time to make it robust, efficient, and inexpensive. As a result, it is natural to use IP fabrics as a transit network for Geneve. Fortunately, the use of IP encapsulation and addressing is enough to achieve the primary goal of delivering packets to the correct point in the network through standard switching and routing.

In addition, nearly all underlay fabrics are designed to exploit parallelism in traffic to spread load across multiple links without introducing reordering in individual flows. These ECMP techniques typically involve parsing and hashing the addresses and port numbers from the packet to select an outgoing link. However, the use of tunnels often results in poor ECMP performance without additional knowledge of the protocol as the encapsulated traffic is hidden from the fabric by design and only tunnel endpoint addresses are available for hashing.

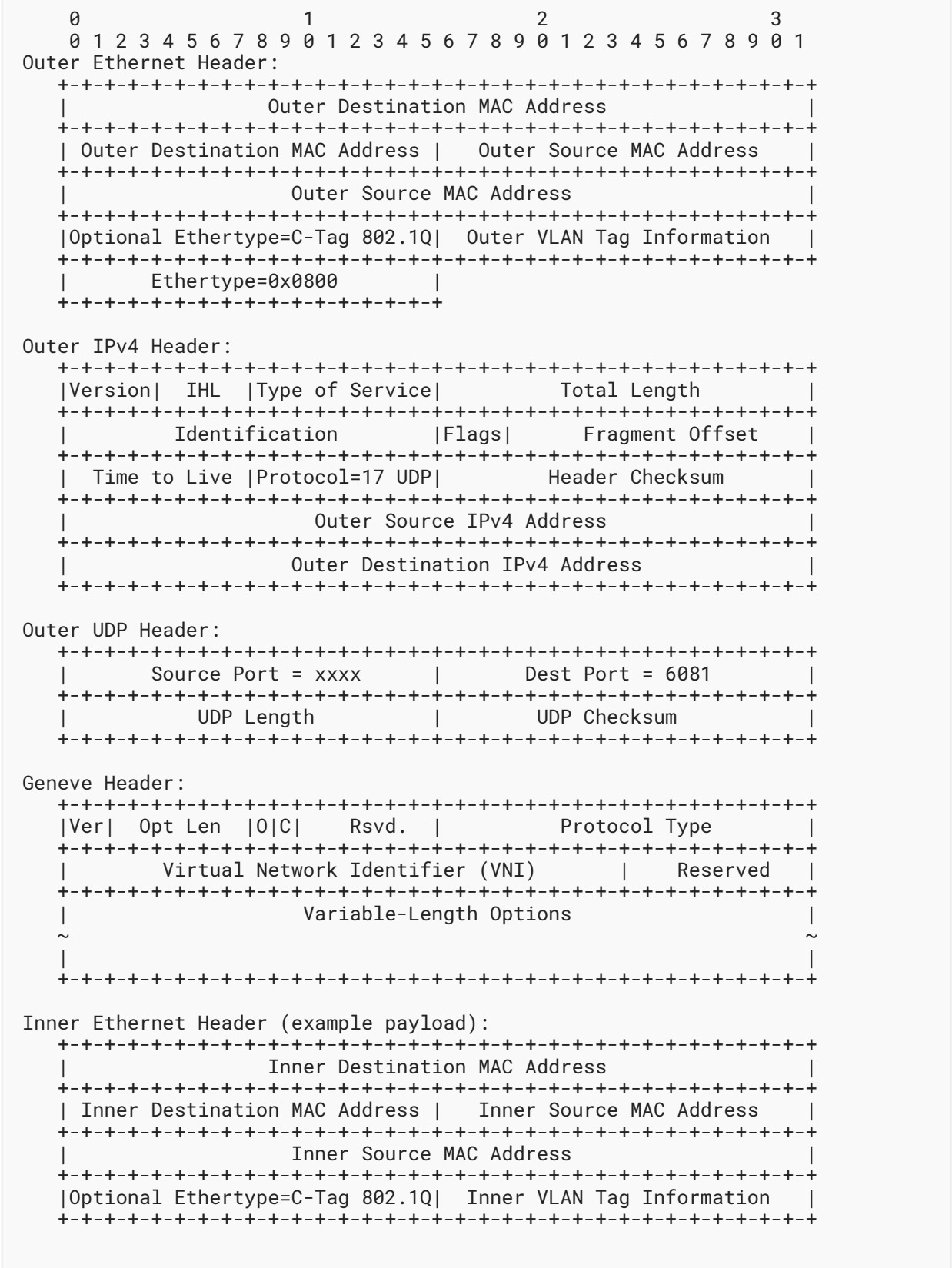
Since it is desirable for Geneve to perform well on these existing fabrics, it is necessary for entropy from encapsulated packets to be exposed in the tunnel header. The most common technique for this is to use the UDP source port, which is discussed further in Section 3.3.

3. Geneve Encapsulation Details

The Geneve packet format consists of a compact tunnel header encapsulated in UDP over either IPv4 or IPv6. A small fixed tunnel header provides control information plus a base level of functionality and interoperability with a focus on simplicity. This header is then followed by a set

of variable options to allow for future innovation. Finally, the payload consists of a protocol data unit of the indicated type, such as an Ethernet frame. Sections 3.1 and 3.2 illustrate the Geneve packet format transported (for example) over Ethernet along with an Ethernet payload.

3.1. Geneve Packet Format over IPv4


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Outer Ethernet Header:
Outer Destination MAC Address
Outer Destination MAC Address
Outer Source MAC Address
Outer Source MAC Address
Optional Ethertype=C-Tag 802.1Q
Outer VLAN Tag Information
Ethertype=0x0800
Outer IPv4 Header:
Version
IHL
Type of Service
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol=17 UDP
Header Checksum
Outer Source IPv4 Address
Outer Destination IPv4 Address
Outer UDP Header:
Source Port = xxxx
Dest Port = 6081
UDP Length
UDP Checksum
Geneve Header:
Ver
Opt Len
O
C
Rsvd.
Protocol Type
Virtual Network Identifier (VNI)
Reserved
Variable-Length Options
Inner Ethernet Header (example payload):
Inner Destination MAC Address
Inner Destination MAC Address
Inner Source MAC Address
Inner Source MAC Address
Optional Ethertype=C-Tag 802.1Q
Inner VLAN Tag Information

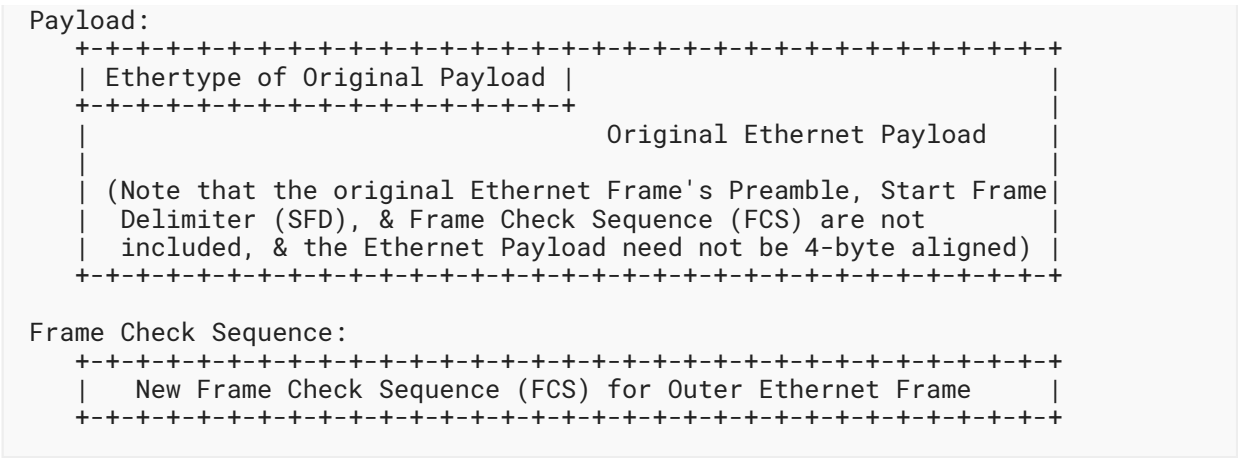
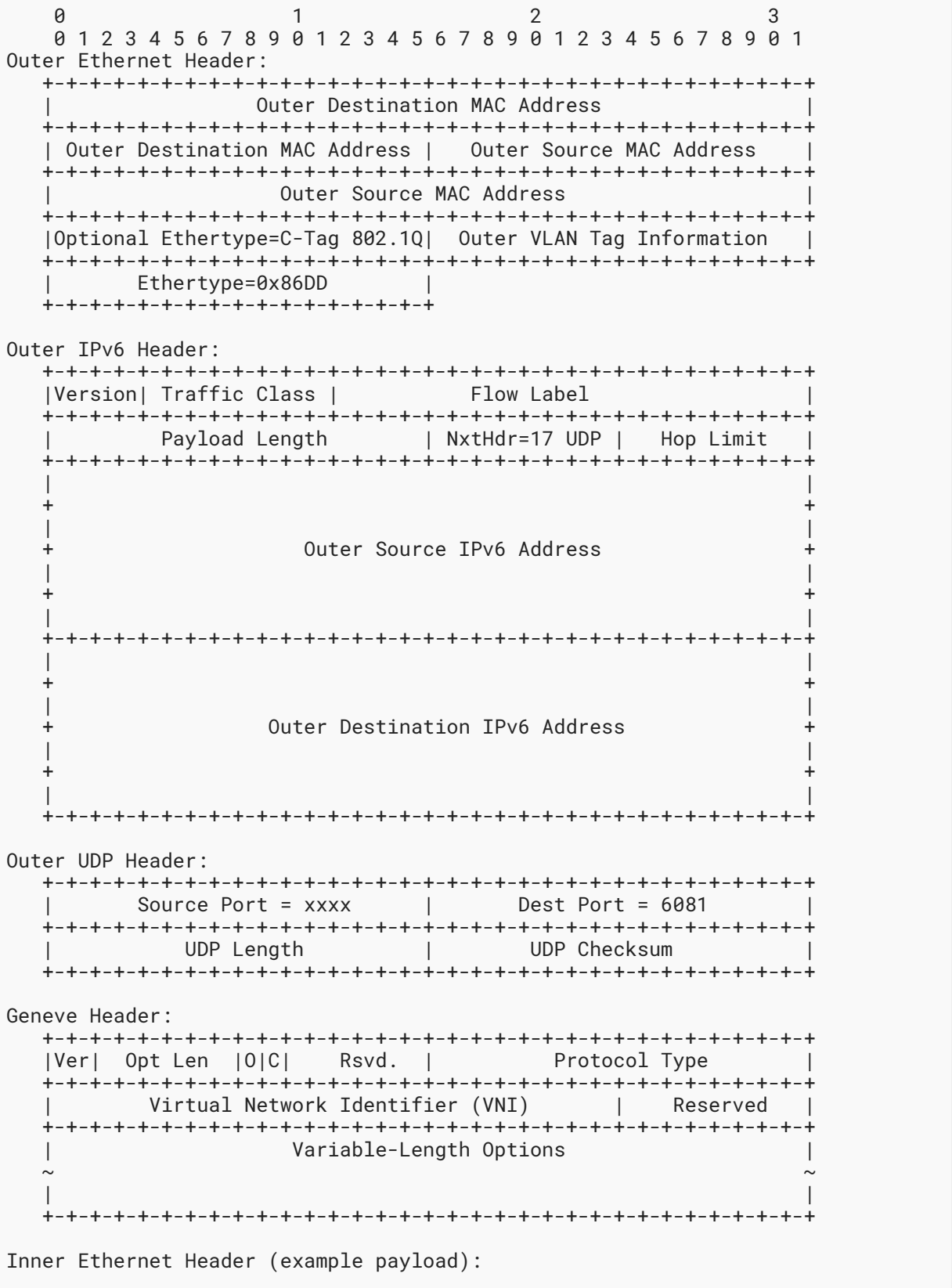


Figure 2: Geneve Packet Format over IPv4

3.2. Geneve Packet Format over IPv6


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Outer Ethernet Header:
Outer Destination MAC Address
Outer Destination MAC Address
Outer Source MAC Address
Outer Source MAC Address
Optional Ethertype=C-Tag 802.1Q
Outer VLAN Tag Information
Ethertype=0x86DD
Outer IPv6 Header:
Version
Traffic Class
Flow Label
Payload Length
NxtHdr=17 UDP
Hop Limit
Outer Source IPv6 Address
Outer Destination IPv6 Address
Outer UDP Header:
Source Port = xxxx
Dest Port = 6081
UDP Length
UDP Checksum
Geneve Header:
Ver
Opt Len
O
C
Rsvd.
Protocol Type
Virtual Network Identifier (VNI)
Reserved
Variable-Length Options
Inner Ethernet Header (example payload):

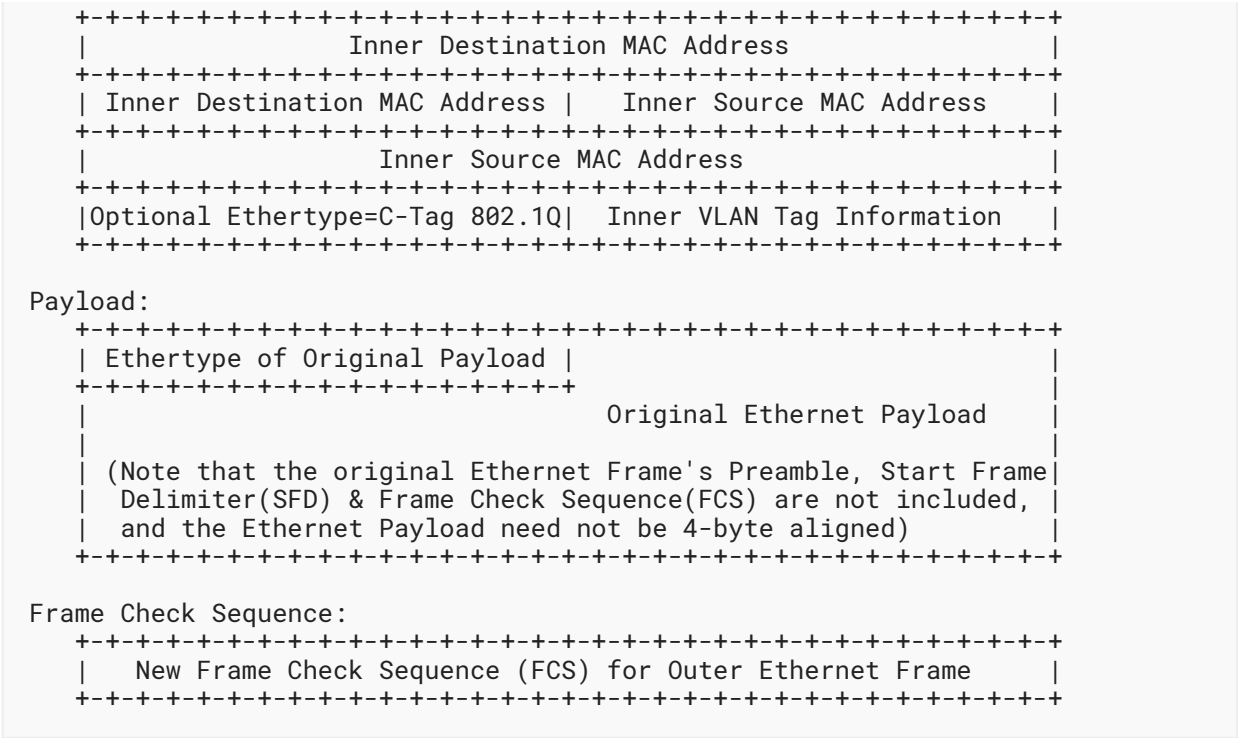


Figure 3: Geneve Packet Format over IPv6

3.3. UDP Header

The use of an encapsulating UDP [RFC0768] header follows the connectionless semantics of Ethernet and IP in addition to providing entropy to routers performing ECMP. Therefore, header fields are interpreted as follows:

Source port: A source port selected by the originating tunnel endpoint. This source port **SHOULD** be the same for all packets belonging to a single encapsulated flow to prevent reordering due to the use of different paths. To encourage an even distribution of flows across multiple links, the source port **SHOULD** be calculated using a hash of the encapsulated packet headers using, for example, a traditional 5-tuple. Since the port represents a flow identifier rather than a true UDP connection, the entire 16-bit range **MAY** be used to maximize entropy. In addition to setting the source port, for IPv6, flow label **MAY** also be used for providing entropy. For an example of using IPv6 flow label for tunnel use cases, see [RFC6438].

If Geneve traffic is shared with other UDP listeners on the same IP address, tunnel endpoints **SHOULD** implement a mechanism to ensure ICMP return traffic arising from network errors is directed to the correct listener. The definition of such a mechanism is beyond the scope of this document.

Dest port: IANA has assigned port 6081 as the fixed well-known destination port for Geneve. Although the well-known value should be used by default, it is **RECOMMENDED** that implementations make this configurable. The chosen port is used for identification of Geneve packets and **MUST NOT** be reversed for different ends of a connection as is done with TCP. It is

the responsibility of the control plane for any reconfiguration of the assigned port and its interpretation by respective devices. The definition of the control plane is beyond the scope of this document.

UDP length: The length of the UDP packet including the UDP header.

UDP checksum: In order to protect the Geneve header, options, and payload from potential data corruption, the UDP checksum **SHOULD** be generated as specified in [RFC0768] and [RFC1112] when Geneve is encapsulated in IPv4. To protect the IP header, Geneve header, options, and payload from potential data corruption, the UDP checksum **MUST** be generated by default as specified in [RFC0768] and [RFC8200] when Geneve is encapsulated in IPv6, except under certain conditions, which are outlined in the next paragraph. Upon receiving such packets with a non-zero UDP checksum, the receiving tunnel endpoints **MUST** validate the checksum. If the checksum is not correct, the packet **MUST** be dropped; otherwise, the packet **MUST** be accepted for decapsulation.

Under certain conditions, the UDP checksum **MAY** be set to zero on transmit for packets encapsulated in both IPv4 and IPv6 [RFC8200]. See Section 4.3 for additional requirements that apply when using zero UDP checksum with IPv4 and IPv6. Disabling the use of UDP checksums is an operational consideration that should take into account the risks and effects of packet corruption.

3.4. Tunnel Header Fields

Ver (2 bits): The current version number is 0. Packets received by a tunnel endpoint with an unknown version **MUST** be dropped. Transit devices interpreting Geneve packets with an unknown version number **MUST** treat them as UDP packets with an unknown payload.

Opt Len (6 bits): The length of the options fields, expressed in 4-byte multiples, not including the eight-byte fixed tunnel header. This results in a minimum total Geneve header size of 8 bytes and a maximum of 260 bytes. The start of the payload headers can be found using this offset from the end of the base Geneve header.

Transit devices **MUST** maintain consistent forwarding behavior irrespective of the value of 'Opt Len', including ECMP link selection.

O (1 bit): Control packet. This packet contains a control message. Control messages are sent between tunnel endpoints. Tunnel endpoints **MUST NOT** forward the payload, and transit devices **MUST NOT** attempt to interpret it. Since control messages are less frequent, it is **RECOMMENDED** that tunnel endpoints direct these packets to a high-priority control queue (for example, to direct the packet to a general purpose CPU from a forwarding Application-Specific Integrated Circuit (ASIC) or to separate out control traffic on a NIC). Transit devices **MUST NOT** alter forwarding behavior on the basis of this bit, such as ECMP link selection.

C (1 bit): Critical options present. One or more options has the critical bit set (see Section 3.5). If this bit is set, then tunnel endpoints **MUST** parse the options list to interpret any critical options. On tunnel endpoints where option parsing is not supported, the packet **MUST** be

dropped on the basis of the 'C' bit in the base header. If the bit is not set, tunnel endpoints **MAY** strip all options using 'Opt Len' and forward the decapsulated packet. Transit devices **MUST NOT** drop packets on the basis of this bit.

Rsvd. (6 bits): Reserved field, which **MUST** be zero on transmission and **MUST** be ignored on receipt.

Protocol Type (16 bits): The type of protocol data unit appearing after the Geneve header. This follows the EtherType [ETYPES] convention, with Ethernet itself being represented by the value 0x6558.

Virtual Network Identifier (VNI) (24 bits): An identifier for a unique element of a virtual network. In many situations, this may represent an L2 segment; however, the control plane defines the forwarding semantics of decapsulated packets. The VNI **MAY** be used as part of ECMP forwarding decisions or **MAY** be used as a mechanism to distinguish between overlapping address spaces contained in the encapsulated packet when load balancing across CPUs.

Reserved (8 bits): Reserved field, which **MUST** be zero on transmission and ignored on receipt.

3.5. Tunnel Options

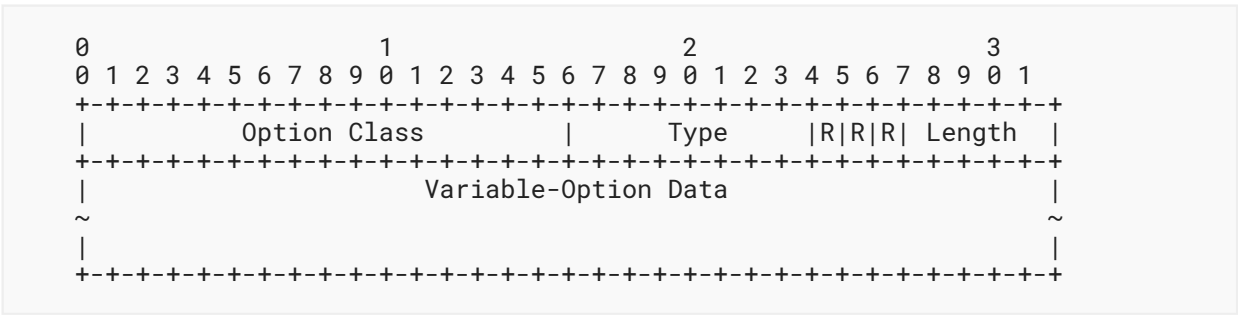


Figure 4: Geneve Option

The base Geneve header is followed by zero or more options in Type-Length-Value format. Each option consists of a 4-byte option header and a variable amount of option data interpreted according to the type.

Option Class (16 bits): Namespace for the 'Type' field. IANA has created a "Geneve Option Class" registry to allocate identifiers for organizations, technologies, and vendors that have an interest in creating types for options. Each organization may allocate types independently to allow experimentation and rapid innovation. It is expected that, over time, certain options will become well known, and a given implementation may use option types from a variety of sources. In addition, IANA has reserved specific ranges for allocation by IETF Review and for Experimental Use (see Section 7).

Type (8 bits): Type indicating the format of the data contained in this option. Options are primarily designed to encourage future extensibility and innovation, and standardized forms of these options will be defined in separate documents.

The high-order bit of the option type indicates that this is a critical option. If the receiving tunnel endpoint does not recognize this option and this bit is set, then the packet **MUST** be dropped. If this bit is set in any option, then the 'C' bit in the Geneve base header **MUST** also be set. Transit devices **MUST NOT** drop packets on the basis of this bit. The following figure shows the location of the 'C' bit in the 'Type' field:

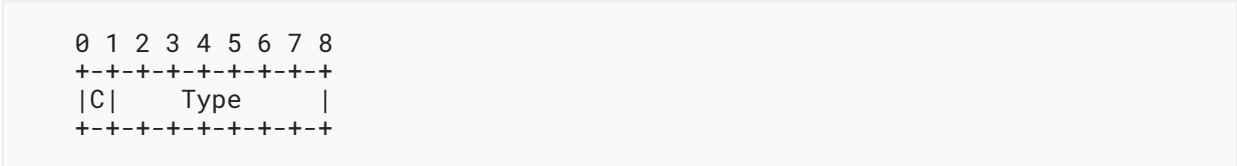


Figure 5: 'C' Bit in the 'Type' Field

The requirement to drop a packet with an unknown option with the 'C' bit set applies to the entire tunnel endpoint system and not a particular component of the implementation. For example, in a system comprised of a forwarding ASIC and a general purpose CPU, this does not mean that the packet must be dropped in the ASIC. An implementation may send the packet to the CPU using a rate-limited control channel for slow-path exception handling.

R (3 bits): Option control flags reserved for future use. These bits **MUST** be zero on transmission and **MUST** be ignored on receipt.

Length (5 bits): Length of the option, expressed in 4-byte multiples excluding the option header. The total length of each option may be between 4 and 128 bytes. A value of 0 in the Length field implies an option with only an option header and no variable-option data. Packets in which the total length of all options is not equal to the 'Opt Len' in the base header are invalid and **MUST** be silently dropped if received by a tunnel endpoint that processes the options.

Variable-Option Data: Option data interpreted according to 'Type'.

3.5.1. Options Processing

Geneve options are intended to be originated and processed by tunnel endpoints. However, options **MAY** be interpreted by transit devices along the tunnel path. Transit devices not interpreting Geneve headers (which may or may not include options) **MUST** handle Geneve packets as any other UDP packet and maintain consistent forwarding behavior.

In tunnel endpoints, the generation and interpretation of options is determined by the control plane, which is beyond the scope of this document. However, to ensure interoperability between heterogeneous devices, some requirements are imposed on options and the devices that process them:

- Receiving tunnel endpoints **MUST** drop packets containing unknown options with the 'C' bit set in the option type. Conversely, transit devices **MUST NOT** drop packets as a result of encountering unknown options, including those with the 'C' bit set.
- The contents of the options and their ordering **MUST NOT** be modified by transit devices.
- If a tunnel endpoint receives a Geneve packet with 'Opt Len' (total length of all options) that exceeds the options-processing capability of the tunnel endpoint, then the tunnel endpoint **MUST** drop such packets. An implementation may raise an exception to the control plane of such an event. It is the responsibility of the control plane to ensure the communicating peer tunnel endpoints have the processing capability to handle the total length of options. The definition of the control plane is beyond the scope of this document.

When designing a Geneve option, it is important to consider how the option will evolve in the future. Once an option is defined, it is reasonable to expect that implementations may come to depend on a specific behavior. As a result, the scope of any future changes must be carefully described upfront.

Architecturally, options are intended to be self descriptive and independent. This enables parallelism in options processing and reduces implementation complexity. However, the control plane may impose certain ordering restrictions, as described in Section 4.5.1.

Unexpectedly significant interoperability issues may result from changing the length of an option that was defined to be a certain size. A particular option is specified to have either a fixed length, which is constant, or a variable length, which may change over time or for different use cases. This property is part of the definition of the option and is conveyed by the 'Type'. For fixed-length options, some implementations may choose to ignore the Length field in the option header and instead parse based on the well-known length associated with the type. In this case, redefining the length will impact not only the parsing of the option in question but also any options that follow. Therefore, options that are defined to be a fixed length in size **MUST NOT** be redefined to a different length. Instead, a new 'Type' should be allocated. Actual definition of the option type is beyond the scope of this document. The option type and its interpretation should be defined by the entity that owns the option class.

Options may be processed by NIC hardware utilizing offloads (e.g., LSO and LRO) as described in Section 4.6. Careful consideration should be given to how the offload capabilities outlined in Section 4.6 impact an option's design.

4. Implementation and Deployment Considerations

4.1. Applicability Statement

Geneve is a network virtualization overlay encapsulation protocol designed to establish tunnels between NVEs over an existing IP network. It is intended for use in public or private data-center environments, for deploying multi-tenant overlay networks over an existing IP underlay network.

Geneve is a UDP-based encapsulation protocol transported over existing IPv4 and IPv6 networks. Hence, as a UDP-based protocol, Geneve adheres to the UDP usage guidelines as specified in [RFC8085]. The applicability of these guidelines are dependent on the underlay IP network and the nature of the Geneve payload protocol (for example, TCP/IP, IP/Ethernet).

Geneve is intended to be deployed in a data-center network environment operated by a single operator or an adjacent set of cooperating network operators that fits with the definition of controlled environments in [RFC8085]. A network in a controlled environment can be managed to operate under certain conditions, whereas in the general Internet, this cannot be done. Hence, requirements for a tunnel protocol operating under a controlled environment can be less restrictive than the requirements of the general Internet.

For the purpose of this document, a traffic-managed controlled environment (TMCE) is defined as an IP network that is traffic engineered and/or otherwise managed (e.g., via use of traffic rate limiters) to avoid congestion. The concept of TMCE is outlined in [RFC8086]. Significant portions of the text in Section 4.1 through Section 4.3 are based on [RFC8086] as applicable to Geneve.

It is the responsibility of the operator to ensure that the guidelines/requirements in this section are followed as applicable to their Geneve deployment(s).

4.2. Congestion-Control Functionality

Geneve does not natively provide congestion control functionality and relies on the payload protocol traffic for congestion control. As such, Geneve **MUST** be used with congestion-controlled traffic or within a network that is traffic managed to avoid congestion (TMCE). An operator of a traffic-managed network (TMCE) may avoid congestion by careful provisioning of their networks, rate-limiting of user data traffic and traffic engineering according to path capacity.

4.3. UDP Checksum

In order to provide integrity of Geneve headers, options and payload (for example, to avoid misdelivery of payload to different tenant systems) in case of data corruption, the outer UDP checksum **SHOULD** be used with Geneve when transported over IPv4. The UDP checksum provides a statistical guarantee that a payload was not corrupted in transit. These integrity checks are not strong from a coding or cryptographic perspective and are not designed to detect

physical-layer errors or malicious modification of the datagram (see Section 3.4 of [RFC8085]). In deployments where such a risk exists, an operator **SHOULD** use additional data integrity mechanisms such as those offered by IPsec (see Section 6.2).

An operator **MAY** choose to disable UDP checksums and use zero checksums if Geneve packet integrity is provided by other data integrity mechanisms, such as IPsec or additional checksums, or if one of the conditions (a, b, or c) in Section 4.3.1 are met.

By default, UDP checksums **MUST** be used when Geneve is transported over IPv6. A tunnel endpoint **MAY** be configured for use with zero UDP checksum if additional requirements in Section 4.3.1 are met.

4.3.1. UDP Zero Checksum Handling with IPv6

When Geneve is used over IPv6, the UDP checksum is used to protect IPv6 headers, UDP headers, and Geneve headers, options, and payload from potential data corruption. As such, by default, Geneve **MUST** use UDP checksums when transported over IPv6. An operator **MAY** choose to configure to operate with zero UDP checksum if operating in a traffic-managed controlled environment as stated in Section 4.1 if one of the following conditions are met.

a. It is known that packet corruption is exceptionally unlikely (perhaps based on knowledge of equipment types in their underlay network) and the operator is willing to risk undetected packet corruption.
b. It is judged through observational measurements (perhaps through historic or current traffic flows that use non-zero checksum) that the level of packet corruption is tolerably low and is where the operator is willing to risk undetected corruption.
c. The Geneve payload is carrying applications that are tolerant of misdelivered or corrupted packets (perhaps through higher-layer checksum validation and/or reliability through retransmission).

In addition, Geneve tunnel implementations using zero UDP checksum **MUST** meet the following requirements:

1. Use of UDP checksum over IPv6 **MUST** be the default configuration for all Geneve tunnels.
2. If Geneve is used with zero UDP checksum over IPv6, then such a tunnel endpoint implementation **MUST** meet all the requirements specified in Section 4 of [RFC6936] and requirement 1 as specified in Section 5 of [RFC6936] as is relevant to Geneve.
3. The Geneve tunnel endpoint that decapsulates the tunnel **SHOULD** check that the source and destination IPv6 addresses are valid for the Geneve tunnel that is configured to receive zero UDP checksum and discard other packets for which such a check fails.
4. The Geneve tunnel endpoint that encapsulates the tunnel **MAY** use different IPv6 source addresses for each Geneve tunnel that uses zero UDP checksum mode in order to strengthen the decapsulator's check of the IPv6 source address (i.e., the same IPv6 source address is not to be used with more than one IPv6 destination address, irrespective of whether that destination address is a unicast or multicast address). When this is not possible, it is **RECOMMENDED** to use each source address for as few Geneve tunnels that use zero UDP checksum as is feasible.

Note that (for requirements 3 and 4) the receiving tunnel endpoint can apply these checks only if it has out-of-band knowledge that the encapsulating tunnel endpoint is applying the indicated behavior. One possibility to obtain this out-of-band knowledge is through signaling by the control plane. The definition of the control plane is beyond the scope of this document.

5. Measures **SHOULD** be taken to prevent Geneve traffic over IPv6 with zero UDP checksum from escaping into the general Internet. Examples of such measures include employing packet filters at the gateways or edge of the Geneve network and/or keeping logical or physical separation of the Geneve network from networks carrying the general Internet traffic.

The above requirements do not change either the requirements specified in [RFC8200] or the requirements specified in [RFC6936].

The use of the source IPv6 address in addition to the destination IPv6 address, plus the recommendation against reuse of source IPv6 addresses among Geneve tunnels, collectively provide some mitigation for the absence of UDP checksum coverage of the IPv6 header. A traffic-managed controlled environment that satisfies at least one of three conditions listed at the beginning of this section provides additional assurance.

4.4. Encapsulation of Geneve in IP

As an IP-based tunnel protocol, Geneve shares many properties and techniques with existing protocols. The application of some of these are described in further detail, although, in general, most concepts applicable to the IP layer or to IP tunnels generally also function in the context of Geneve.

4.4.1. IP Fragmentation

It is strongly **RECOMMENDED** that Path MTU Discovery ([RFC1191] [RFC8201]) be used to prevent or minimize fragmentation. The use of Path MTU Discovery on the transit network provides the encapsulating tunnel endpoint with soft-state about the link that it may use to prevent or minimize fragmentation depending on its role in the virtualized network. The NVE can maintain this state (the MTU size of the tunnel link(s) associated with the tunnel endpoint), so if a tenant system sends large packets that, when encapsulated, exceed the MTU size of the tunnel link, the tunnel endpoint can discard such packets and send exception messages to the tenant system(s). If the tunnel endpoint is associated with a routing or forwarding function and/or has the capability to send ICMP messages, the encapsulating tunnel endpoint **MAY** send ICMP fragmentation needed [RFC0792] or Packet Too Big [RFC4443] messages to the tenant system(s). When determining the MTU size of a tunnel link, the maximum length of options **MUST** be assumed as options may vary on a per-packet basis. For example, recommendations/guidance for handling fragmentation in similar overlay encapsulation services like Pseudowire Emulation Edge-to-Edge (PWE3) are provided in Section 5.3 of [RFC3985].

Note that some implementations may not be capable of supporting fragmentation or other less common features of the IP header, such as options and extension headers. For example, some of the issues associated with MTU size and fragmentation in IP tunneling and use of ICMP messages are outlined in Section 4.2 of [INTAREA-TUNNELS].

4.4.2. DSCP, ECN, and TTL

When encapsulating IP (including over Ethernet) packets in Geneve, there are several considerations for propagating Differentiated Services Code Point (DSCP) and Explicit Congestion Notification (ECN) bits from the inner header to the tunnel on transmission and the reverse on reception.

[RFC2983] provides guidance for mapping DSCP between inner and outer IP headers. Network virtualization is typically more closely aligned with the Pipe model described, where the DSCP value on the tunnel header is set based on a policy (which may be a fixed value, one based on the inner traffic class or some other mechanism for grouping traffic). Aspects of the Uniform model (which treats the inner and outer DSCP values as a single field by copying on ingress and egress) may also apply, such as the ability to re-mark the inner header on tunnel egress based on transit marking. However, the Uniform model is not conceptually consistent with network virtualization, which seeks to provide strong isolation between encapsulated traffic and the physical network.

[RFC6040] describes the mechanism for exposing ECN capabilities on IP tunnels and propagating congestion markers to the inner packets. This behavior **MUST** be followed for IP packets encapsulated in Geneve.

Though Uniform or Pipe models could be used for handling TTL (or Hop Limit in case of IPv6) when tunneling IP packets, the Pipe model is more aligned with network virtualization. [RFC2003] provides guidance on handling TTL between inner IP header and outer IP tunnels; this model is more aligned with the Pipe model and is **RECOMMENDED** for use with Geneve for network virtualization applications.

4.4.3. Broadcast and Multicast

Geneve tunnels may either be point-to-point unicast between two tunnel endpoints or utilize broadcast or multicast addressing. It is not required that inner and outer addressing match in this respect. For example, in physical networks that do not support multicast, encapsulated multicast traffic may be replicated into multiple unicast tunnels or forwarded by policy to a unicast location (possibly to be replicated there).

With physical networks that do support multicast, it may be desirable to use this capability to take advantage of hardware replication for encapsulated packets. In this case, multicast addresses may be allocated in the physical network corresponding to tenants, encapsulated multicast groups, or some other factor. The allocation of these groups is a component of the control plane and, therefore, is beyond the scope of this document.

When physical multicast is in use, devices with heterogeneous capabilities may be present in the same group. Some options may only be interpretable by a subset of the devices in the group. Other devices can safely ignore such options unless the 'C' bit is set to mark the unknown option as critical. The requirements outlined in Section 3.4 apply for critical options.

In addition, [RFC8293] provides examples of various mechanisms that can be used for multicast handling in network virtualization overlay networks.

4.4.4. Unidirectional Tunnels

Generally speaking, a Geneve tunnel is a unidirectional concept. IP is not a connection-oriented protocol, and it is possible for two tunnel endpoints to communicate with each other using different paths or to have one side not transmit anything at all. As Geneve is an IP-based protocol, the tunnel layer inherits these same characteristics.

It is possible for a tunnel to encapsulate a protocol, such as TCP, that is connection oriented and maintains session state at that layer. In addition, implementations **MAY** model Geneve tunnels as connected, bidirectional links, for example, to provide the abstraction of a virtual port. In both of these cases, bidirectionality of the tunnel is handled at a higher layer and does not affect the operation of Geneve itself.

4.5. Constraints on Protocol Features

Geneve is intended to be flexible to a wide range of current and future applications. As a result, certain constraints may be placed on the use of metadata or other aspects of the protocol in order to optimize for a particular use case. For example, some applications may limit the types of options that are supported or enforce a maximum number or length of options. Other applications may only handle certain encapsulated payload types, such as Ethernet or IP. This could be either globally throughout the system or, for example, restricted to certain classes of devices or network paths.

These constraints may be communicated to tunnel endpoints either explicitly through a control plane or implicitly by the nature of the application. As Geneve is defined as a data-plane protocol that is control plane agnostic, definition of such mechanisms are beyond the scope of this document.

4.5.1. Constraints on Options

While Geneve options are flexible, a control plane may restrict the number of option TLVs as well as the order and size of the TLVs between tunnel endpoints to make it simpler for a data-plane implementation in software or hardware to handle [NVO3-ENCAP]. For example, there may be some critical information, such as a secure hash, that must be processed in a certain order to provide the lowest latency, or there may be other scenarios where the options must be processed in a certain order due to protocol semantics.

A control plane may negotiate a subset of option TLVs and certain TLV ordering; it may also limit the total number of option TLVs present in the packet, for example, to accommodate hardware capable of processing fewer options [NVO3-ENCAP]. Hence, a control plane needs to have the

ability to describe the supported TLVs subset and their order to the tunnel endpoints. In the absence of a control plane, alternative configuration mechanisms may be used for this purpose. Such mechanisms are beyond the scope of this document.

4.6. NIC Offloads

Modern NICs currently provide a variety of offloads to enable the efficient processing of packets. The implementation of many of these offloads requires only that the encapsulated packet be easily parsed (for example, checksum offload). However, optimizations such as LSO and LRO involve some processing of the options themselves since they must be replicated/merged across multiple packets. In these situations, it is desirable not to require changes to the offload logic to handle the introduction of new options. To enable this, some constraints are placed on the definitions of options to allow for simple processing rules:

- When performing LSO, a NIC **MUST** replicate the entire Geneve header and all options, including those unknown to the device, onto each resulting segment unless an option allows an exception. Conversely, when performing LRO, a NIC may assume that a binary comparison of the options (including unknown options) is sufficient to ensure equality and **MAY** merge packets with equal Geneve headers.
- Options **MUST NOT** be reordered during the course of offload processing, including when merging packets for the purpose of LRO.
- NICs performing offloads **MUST NOT** drop packets with unknown options, including those marked as critical, unless explicitly configured.

There is no requirement that a given implementation of Geneve employ the offloads listed as examples above. However, as these offloads are currently widely deployed in commercially available NICs, the rules described here are intended to enable efficient handling of current and future options across a variety of devices.

4.7. Inner VLAN Handling

Geneve is capable of encapsulating a wide range of protocols; therefore, a given implementation is likely to support only a small subset of the possibilities. However, as Ethernet is expected to be widely deployed, it is useful to describe the behavior of VLANs inside encapsulated Ethernet frames.

As with any protocol, support for inner VLAN headers is **OPTIONAL**. In many cases, the use of encapsulated VLANs may be disallowed due to security or implementation considerations. However, in other cases, the trunking of VLAN frames across a Geneve tunnel can prove useful. As a result, the processing of inner VLAN tags upon ingress or egress from a tunnel endpoint is based upon the configuration of the tunnel endpoint and/or control plane and is not explicitly defined as part of the data format.

5. Transition Considerations

Viewed exclusively from the data plane, Geneve is compatible with existing IP networks as it appears to most devices as UDP packets. However, as there are already a number of tunnel protocols deployed in network virtualization environments, there is a practical question of transition and coexistence.

Since Geneve builds on the base data-plane functionality provided by the most common protocols used for network virtualization (VXLAN, NVGRE), it should be straightforward to port an existing control plane to run on top of it with minimal effort. With both the old and new packet formats supporting the same set of capabilities, there is no need for a hard transition; tunnel endpoints directly communicating with each other can use any common protocol, which may be different even within a single overall system. As transit devices are primarily forwarding packets on the basis of the IP header, all protocols appear to be similar, and these devices do not introduce additional interoperability concerns.

To assist with this transition, it is strongly suggested that implementations support simultaneous operation of both Geneve and existing tunnel protocols, as it is expected to be common for a single node to communicate with a mixture of other nodes. Eventually, older protocols may be phased out as they are no longer in use.

6. Security Considerations

As it is encapsulated within a UDP/IP packet, Geneve does not have any inherent security mechanisms. As a result, an attacker with access to the underlay network transporting the IP packets has the ability to snoop, alter, or inject packets. Compromised tunnel endpoints or transit devices may also spoof identifiers in the tunnel header to gain access to networks owned by other tenants.

Within a particular security domain, such as a data center operated by a single service provider, the most common and highest-performing security mechanism is isolation of trusted components. Tunnel traffic can be carried over a separate VLAN and filtered at any untrusted boundaries.

When crossing an untrusted link, such as the general Internet, VPN technologies such as IPsec [RFC4301] should be used to provide authentication and/or encryption of the IP packets formed as part of Geneve encapsulation (see Section 6.1.1).

Geneve does not otherwise affect the security of the encapsulated packets. As per the guidelines of BCP 72 [RFC3552], the following sections describe potential security risks that may be applicable to Geneve deployments and approaches to mitigate such risks. It is also noted that not all such risks are applicable to all Geneve deployment scenarios, i.e., only a subset may be applicable to certain deployments. So an operator has to make an assessment based on their network environment, determine the risks that are applicable to their specific environment, and use appropriate mitigation approaches as applicable.

6.1. Data Confidentiality

Geneve is a network virtualization overlay encapsulation protocol designed to establish tunnels between NVEs over an existing IP network. It can be used to deploy multi-tenant overlay networks over an existing IP underlay network in a public or private data center. The overlay service is typically provided by a service provider, such as a cloud service provider or a private data-center operator. This may or not may be the same provider as an underlay service provider. Due to the nature of multi-tenancy in such environments, a tenant system may expect data confidentiality to ensure its packet data is not tampered with (i.e., active attack) in transit or is a target of unauthorized monitoring (i.e., passive attack), for example, by other tenant systems or underlay service provider. A compromised network node or a transit device within a data center may passively monitor Geneve packet data between NVEs or route traffic for further inspection. A tenant may expect the overlay service provider to provide data confidentiality as part of the service, or a tenant may bring its own data confidentiality mechanisms like IPsec or TLS to protect the data end to end between its tenant systems. The overlay provider is expected to provide cryptographic protection in cases where the underlay provider is not the same as the overlay provider to ensure the payload is not exposed to the underlay.

If an operator determines data confidentiality is necessary in their environment based on their risk analysis -- for example, in multi-tenant environments -- then an encryption mechanism **SHOULD** be used to encrypt the tenant data end to end between the NVEs. The NVEs may use existing well-established encryption mechanisms, such as IPsec, DTLS, etc.

6.1.1. Inter-Data-Center Traffic

A tenant system in a customer premises (private data center) may want to connect to tenant systems on their tenant overlay network in a public cloud data center, or a tenant may want to have its tenant systems located in multiple geographically separated data centers for high availability. Geneve data traffic between tenant systems across such separated networks should be protected from threats when traversing public networks. Any Geneve overlay data leaving the data-center network beyond the operator's security domain **SHOULD** be secured by encryption mechanisms, such as IPsec or other VPN technologies, to protect the communications between the NVEs when they are geographically separated over untrusted network links. Specification of data protection mechanisms employed between data centers is beyond the scope of this document.

The principles described in Section 4 regarding controlled environments still apply to the geographically separated data-center usage outlined in this section.

6.2. Data Integrity

Geneve encapsulation is used between NVEs to establish overlay tunnels over an existing IP underlay network. In a multi-tenant data center, a rogue or compromised tenant system may try to launch a passive attack, such as monitoring the traffic of other tenants, or an active attack, such as trying to inject unauthorized Geneve encapsulated traffic such as spoofing, replay, etc., into the network. To prevent such attacks, an NVE **MUST NOT** propagate Geneve packets beyond

the NVE to tenant systems and **SHOULD** employ packet-filtering mechanisms so as not to forward unauthorized traffic between tenant systems in different tenant networks. An NVE **MUST NOT** interpret Geneve packets from tenant systems other than as frames to be encapsulated.

A compromised network node or a transit device within a data center may launch an active attack trying to tamper with the Geneve packet data between NVEs. Malicious tampering of Geneve header fields may cause the packet from one tenant to be forwarded to a different tenant network. If an operator determines the possibility of such a threat in their environment, the operator may choose to employ data integrity mechanisms between NVEs. In order to prevent such risks, a data integrity mechanism **SHOULD** be used in such environments to protect the integrity of Geneve packets, including packet headers, options, and payload on communications between NVE pairs. A cryptographic data protection mechanism, such as IPsec, may be used to provide data integrity protection. A data-center operator may choose to deploy any other data integrity mechanisms as applicable and supported in their underlay networks, although non-cryptographic mechanisms may not protect the Geneve portion of the packet from tampering.

6.3. Authentication of NVE Peers

A rogue network device or a compromised NVE in a data-center environment might be able to spoof Geneve packets as if it came from a legitimate NVE. In order to mitigate such a risk, an operator **SHOULD** use an authentication mechanism, such as IPsec, to ensure that the Geneve packet originated from the intended NVE peer in environments where the operator determines spoofing or rogue devices are potential threats. Other simpler source checks, such as ingress filtering for VLAN/MAC/IP addresses, reverse path forwarding checks, etc., may be used in certain trusted environments to ensure Geneve packets originated from the intended NVE peer.

6.4. Options Interpretation by Transit Devices

Options, if present in the packet, are generated and terminated by tunnel endpoints. As indicated in Section 2.2.1, transit devices may interpret the options. However, if the packet is protected by encryption from tunnel endpoint to tunnel endpoint (for example, through IPsec), transit devices will not have visibility into the Geneve header or options in the packet. In such cases, transit devices **MUST** handle Geneve packets as any other IP packet and maintain consistent forwarding behavior. In cases where options are interpreted by transit devices, the operator **MUST** ensure that transit devices are trusted and not compromised. The definition of a mechanism to ensure this trust is beyond the scope of this document.

6.5. Multicast/Broadcast

In typical data-center networks where IP multicasting is not supported in the underlay network, multicasting may be supported using multiple unicast tunnels. The same security requirements as described in the above sections can be used to protect Geneve communications between NVE peers. If IP multicasting is supported in the underlay network and the operator chooses to use it for multicast traffic among tunnel endpoints, then the operator in such environments may use data protection mechanisms, such as IPsec with multicast extensions [RFC5374], to protect multicast traffic among Geneve NVE groups.

6.6. Control-Plane Communications

A Network Virtualization Authority (NVA) as outlined in [RFC8014] may be used as a control plane for configuring and managing the Geneve NVEs. The data-center operator is expected to use security mechanisms to protect the communications between the NVA to NVEs and to use authentication mechanisms to detect any rogue or compromised NVEs within their administrative domain. Data protection mechanisms for control-plane communication or authentication mechanisms between the NVA and the NVEs are beyond the scope of this document.

7. IANA Considerations

IANA has allocated UDP port 6081 in the "Service Name and Transport Protocol Port Number Registry" [IANA-SN] as the well-known destination port for Geneve.

This registration references this document (RFC 8926); in line with [RFC6335], the assignee and contact of the port entry are IESG <iesg@ietf.org> and IETF Chair <chair@ietf.org>, respectively:

Service Name: geneve
Transport Protocol(s): UDP
Assignee: IESG <iesg@ietf.org>
Contact: IETF Chair <chair@ietf.org>
Description: Generic Network Virtualization Encapsulation (Geneve)
Reference: [RFC8926]
Port Number: 6081

In addition, IANA has created a new subregistry title "Geneve Option Class" for Option Classes. This registry has been placed under a new "Network Virtualization Overlay (NVO3)" heading in IANA protocol registries [IANA-PR]. The "Geneve Option Class" registry consists of 16-bit hexadecimal values along with descriptive strings, assignee/contact information, and references. The registration rules for the new registry are (as defined by [RFC8126]):

Range	Registration Procedures
0x0000-0x00FF	IETF Review
0x0100-0xFEFF	First Come First Served
0xFF00-0xFFFF	Experimental Use

Table 1: Geneve Option Class Registry Ranges

Initial registrations in the new registry are as follows:

Option Class	Description	Assignee/Contact	Reference
0x0100	Linux		
0x0101	Open vSwitch (OVS)		
0x0102	Open Virtual Networking (OVN)		
0x0103	In-band Network Telemetry (INT)		
0x0104	VMware, Inc.		
0x0105	Amazon.com, Inc.		
0x0106	Cisco Systems, Inc.		
0x0107	Oracle Corporation		
0x0108-0x0110	Amazon.com, Inc.		

Table 2: Geneve Option Class Registry

8. References

8.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <https://www.rfc-editor.org/info/rfc1112>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <https://www.rfc-editor.org/info/rfc2003>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <https://www.rfc-editor.org/info/rfc6040>.

[RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <https://www.rfc-editor.org/info/rfc6936>.

[RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for Data Center (DC) Network Virtualization", RFC 7365, DOI 10.17487/RFC7365, October 2014, <https://www.rfc-editor.org/info/rfc7365>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

8.2. Informative References

[ETYPES] IANA, "IEEE 802 Numbers", <https://www.iana.org/assignments/ieee-802-numbers>.

[IANA-PR] IANA, "Protocol Registries", <https://www.iana.org/protocols>.

[IANA-SN] IANA, "Service Name and Transport Protocol Port Number Registry", <https://www.iana.org/assignments/service-names-port-numbers>.

[IEEE.802.1Q_2018] IEEE, "IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks", DOI 10.1109/IEEESTD.2018.8403927, IEEE 802.1Q-2018, July 2018, <http://ieeexplore.ieee.org/servlet/opac?punumber=8403925>.

[INTAREA-TUNNELS] Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", Work in Progress, Internet-Draft, draft-ietf-intarea-tunnels-10, 12 September 2019, <https://tools.ietf.org/html/draft-ietf-intarea-tunnels-10>.

[NVO3-DATAPLANE] Bitar, N., Lasserre, M., Balus, F., Morin, T., Jin, L., and B. Khasnabish, "NVO3 Data Plane Requirements", Work in Progress, Internet-Draft, draft-ietf-nvo3-dataplane-requirements-03, 15 April 2014, <https://tools.ietf.org/html/draft-ietf-nvo3-dataplane-requirements-03>.

[NVO3-ENCAP] Boutros, S., "NVO3 Encapsulation Considerations", Work in Progress, Internet-Draft, draft-ietf-nvo3-encap-05, 17 February 2020, <https://tools.ietf.org/html/draft-ietf-nvo3-encap-05>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <https://www.rfc-editor.org/info/rfc2983>.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <https://www.rfc-editor.org/info/rfc3031>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[RFC3985] Bryant, S., Ed. and P. Pate, Ed., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, DOI 10.17487/RFC3985, March 2005, <https://www.rfc-editor.org/info/rfc3985>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC5374] Weis, B., Gross, G., and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol", RFC 5374, DOI 10.17487/RFC5374, November 2008, <https://www.rfc-editor.org/info/rfc5374>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <https://www.rfc-editor.org/info/rfc6438>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <https://www.rfc-editor.org/info/rfc7348>.

[RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <https://www.rfc-editor.org/info/rfc7637>.

[RFC8014] Black, D., Hudson, J., Kreeger, L., Lasserre, M., and T. Narten, "An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)", RFC 8014, DOI 10.17487/RFC8014, December 2016, <https://www.rfc-editor.org/info/rfc8014>.

[RFC8086] Yong, L., Ed., Crabbe, E., Xu, X., and T. Herbert, "GRE-in-UDP Encapsulation", RFC 8086, DOI 10.17487/RFC8086, March 2017, <https://www.rfc-editor.org/info/rfc8086>.

[RFC8293] Ghanwani, A., Dunbar, L., McBride, M., Bannai, V., and R. Krishnan, "A Framework for Multicast in Network Virtualization over Layer 3", RFC 8293, DOI 10.17487/RFC8293, January 2018, <https://www.rfc-editor.org/info/rfc8293>.

[VL2] "VL2: A Scalable and Flexible Data Center Network", ACM SIGCOMM Computer Communication Review, DOI 10.1145/1594977.1592576, August 2009, <https://www.sigcomm.org/sites/default/files/ccr/papers/2009/October/1594977-1592576.pdf>.

Acknowledgements

The authors wish to acknowledge Puneet Agarwal, David Black, Sami Boutros, Scott Bradner, Martin Casado, Alissa Cooper, Roman Danyliw, Bruce Davie, Anoop Ghanwani, Benjamin Kaduk, Suresh Krishnan, Mirja Kühlewind, Barry Leiba, Daniel Migault, Greg Mirksy, Tal Mizrahi, Kathleen Moriarty, Magnus Nystrom, Adam Roach, Sabrina Tanamal, Dave Thaler, Eric Vyncke, Magnus Westerlund, and many other members of the NVO3 WG for their reviews, comments, and suggestions.

The authors would like to thank Sam Aldrin, Alia Atlas, Matthew Bocci, Benson Schliesser, and Martin Vigoureux for their guidance throughout the process.

Contributors

The following individuals were authors of an earlier version of this document and made significant contributions:

Pankaj Garg
Microsoft Corporation
1 Microsoft Way
Redmond, WA 98052
United States of America
Email: pankajg@microsoft.com

Chris Wright
Red Hat Inc.
1801 Varsity Drive
Raleigh, NC 27606
United States of America

Kenneth Duda
Arista Networks
5453 Great America Parkway
Santa Clara, CA 95054
United States of America
Email: kduda@arista.com

Dinesh G. Dutt
Independent
Email: didutt@gmail.com

Jon Hudson
Independent
Email: jon.hudson@gmail.com

Ariel Hendel
Facebook, Inc.
1 Hacker Way
Menlo Park, CA 94025
United States of America
Email: ahendel@fb.com

Authors' Addresses

Jesse Gross (EDITOR)
Email: jesse@kernel.org

Ilango Ganga (EDITOR)
Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054
United States of America
Email: ilango.s.ganga@intel.com

T. Sridhar (EDITOR)
VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
United States of America
Email: tsridhar@vmware.com